



TITLE:

# k-辺連結あるいはk-点連結全域部分グラフについて(アルゴリズムと計算量の理論)

AUTHOR(S):

永持, 仁; 茨木, 俊秀

---

CITATION:

永持, 仁 ...[et al]. k-辺連結あるいはk-点連結全域部分グラフについて(アルゴリズムと計算量の理論). 数理解析研究所講究録 1990, 731: 202-213

ISSUE DATE:

1990-10

URL:

<http://hdl.handle.net/2433/101965>

RIGHT:

**k-辺連結あるいはk-点連結全域部分グラフについて**

豊橋技術科学大学 永持 仁 (Hiroshi Nagamochi)

京都大学 茨木 俊秀 (Toshihide Ibaraki)

**1. はじめに**

与えられた無向グラフ  $G=(V, E)$  および正整数  $k (\geq 2)$  に対し、辺数を最小にするような  $G$  の  $k$ -辺連結全域部分グラフあるいは  $k$ -点連結全域部分グラフを求める問題は、 $k$  を固定しても NP-完全である [7]. 最近、鈴木、高橋、西関らは辺数が高々  $O(k|V|)$  本であるような  $k$ -辺連結 (あるいは  $k$ -点連結) 全域部分グラフを線形時間で見出す問題の重要性を指摘し、 $k \leq 3$  のときそのような  $k$ -点連結全域部分グラフを見出す線形時間アルゴリズムを与えた [21]. 一般の  $k$  に対しては、そのような全域部分グラフは多項式時間で見出せることが示されている. 例えば、辺数が  $O(k|V|)$  本である  $k$ -辺連結あるいは  $k$ -点連結全域部分グラフはそれぞれ  $O(k|E|)$ ,  $O(|V|^{1/2}|E|^2)$  時間で求まることが知られている [15].

本報告で述べるアルゴリズム (FOREST) は、グラフ探索を1度行うだけで、辺数が  $k|V|$  本以下である  $k$ -辺連結 (あるいは  $k$ -点連結) 全域部分グラフを  $O(|V|+|E|)$  時間で見出す手続きである. 従って、このアルゴリズムを2点間の辺素あるいは内素なパスを最大本数見つける問題やグラフの連結度を判定する問題を解くアルゴリズムの前処理として用いれば、必要な計算時間を改善することができる.

次に、このアルゴリズム FOREST の持つ幾つかの有用な性質を利用して、多重グラフの辺連結度  $\lambda(G)$ 、容量付き無向ネットワークの最小カット値、グラフの最小カットの総数などを求める問題を解く高速なアルゴリズムを開発する.

**2. 定義**

$G=(V, E)$  を節点集合  $V$ 、辺集合  $E$  からなるループを持たない無向グラフとする. グラフの2節点間に辺を多重に持つグラフは多重であると呼ばれ、2節点  $u, v$  を端点とする辺集合を  $E_{uv}$  で表わす. ただし、混同の恐れのないときは  $e \in E_{uv}$  を  $e=(u, v)$  でも記す. 多重辺を持たないグラフは単純と呼ばれる.  $G=(V, E)$  から  $E' (\subseteq E)$  の全ての辺を除去して得られるグラフを  $G-E'$  と書く. 同様に、 $G=(V, E)$  から  $V' (\subset V)$  の全ての節点 (および  $V$  に接続している辺) を除去して得られるグラフを  $G-V'$  と書く.

グラフ  $G$  のどの  $k-1$  本の辺を取り除いても  $G$  が非連結にならないとき、 $G$  は  $k$ -辺連結 であるという. グラフ  $G$  の辺連結度  $\lambda(G)$  とは、 $G$  が  $k$ -辺連結であって  $k+1$ -辺連結でないとき  $\lambda(G)=k$  と定義される. グラフ  $G=(V, E)$  の局所辺連結度  $\lambda(x, y; G)$  とは、 $G-E'$  において  $x$  と  $y$  が非連結であるような  $E' (\subseteq E)$  の位数の最小値である. 明らかに、 $\lambda(G)=\min\{\lambda(x, y; G) \mid x, y \in V\}$  が成り立つ.

グラフ  $G$  が  $k+1$  個以上の節点を持ちどの  $k-1$  個の節点を取り除いても非連結にな

らないとき,  $G$  は  $k$ -点連結であるという (例えば, 完全グラフ  $K_{k+1}$  は  $k$ -点連結).  
 グラフ  $G$  の 点連結度  $\kappa(G)$  とは,  $G$  が  $k$ -点連結であって  $k+1$ -点連結でないとき  
 $\kappa(G)=k$  と定義される. 局所点連結度  $\kappa(x, y; G)$  とは, 隣接していない  $x, y$  に対し,  $G-V'$  において  $x$  と  $y$  が非連結となるような  $V' (\subseteq V)$  の位数の最小値である. 隣接する  $x, y$  に対しても  $\kappa(x, y; G) \equiv |V|-1$  と定めておけば,  $\kappa(G) = \min\{\kappa(x, y; G) \mid x, y \in V\}$  が成り立つ (実は,  $\overline{\kappa}(x, y; G)$  を  $x, y$  間の内素なパスの最大本数と定義しても,  
 $\kappa(G) = \min\{\overline{\kappa}(x, y; G) \mid x, y \in V\}$  が成立する).

グラフ  $G=(V, E)$  に対して,  $V'=V, E' \subseteq E$  を満たすグラフ  $G'=(V', E')$  は  $G$  の 全域部分グラフ と呼ばれる. 特に,  $G$  (連結でなくてもよい) の各連結成分上で,  $G'$  の作る部分グラフが全て連結であるとき,  $G'$  は 極大 であるという. 閉路を持たないグラフ  $T$  を 森 と呼び,  $T$  が連結のとき 木 と呼ばれる.

グラフ  $G$  の 2 節点  $x, y$  を 1 点に縮約して得られるグラフを  $G/\{x, y\}$  で表わす (ただし, 発生したループは除去してあるとする). 部分節点集合  $X \subseteq V$  に対して,  $E(X) \equiv \{e \in E \mid e \text{ の両端点の一方が } X \text{ に, もう一方が } V-X \text{ に含まれる}\}$  と定める.  $E(\{x\})$  は  $E(x)$  とも記す.  $|E(x)|$  は節点  $x$  の次数である.

グラフ  $G$  の 節点次数の最小値 を  $\delta(G)$  と書く.

連結無向グラフ  $G=(V, E)$  において,  $G-F$  が非連結となるような部分辺集合  $F \subseteq E$  を  $G$  の カット と呼ぼう. 特に,  $|F|=k$  なるカット  $F$  の集合を  $C^k(G)$  と記す.

同様に, 2 点  $s, t$  が  $G-F$  において非連結となるカット  $F$  を  $s$ - $t$  カット と呼び,  $|F|=k$  なる  $s$ - $t$  カット  $F$  の集合を  $C^k(s, t; G)$  で表わす.

図 1 に  $\delta(G)=7, \lambda(G)=6$  なる多重グラフ  $G$  の例を示しておく.

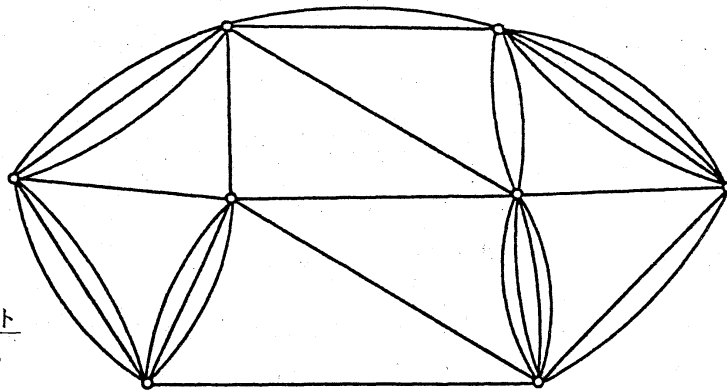


図 1. 多重グラフの例.

### 3. $k$ -辺連結全域部分グラフを求めるアルゴリズム

まず, アルゴリズムの基礎となる定理を示す.

【定理 3.1】 [12] 連結グラフ  $G=(V, E)$  に対し,  $T_1=(V, E_1)$  を  $G$  の任意の極大全域木とし, 以下,  $T_i=(V, E_i), i=1, 2, \dots, k$  を  $G-E_1 \cup E_2 \cup \dots \cup E_{i-1}$  の極大全域森とする. このとき, 各  $G_i=(V, E_1 \cup E_2 \cup \dots \cup E_i), i=1, 2, \dots, |E|$  は, 任意の  $x, y \in V$  に対し,  
 $\lambda(x, y; G_i) \geq \min\{i, \lambda(x, y; G)\}$  を満たす.  $\square$

この定理によれば, 任意の  $\lambda(G) \geq k$  に対し  $G_k=(V, E_1 \cup E_2 \cup \dots \cup E_k)$  は  $k$ -辺連結であり, 各極大全域森  $T_i$  は通常のグラフ探索を用いて  $O(|V|+|E|)$  時間で求まることから, 高々  $k(|V|-1)$  本の辺を持つ  $G$  の  $k$ -辺連結全域部分グラフ  $G'$  は  $O(k|E|)$  時間で得られることが分かる. ところで,  $T_1=(V, E_1)$  を求める最初のグラフ探索において,  $E_1$  に含まれないと判

断された辺に対しても、その辺が含まれるべき $E_i$ が $O(1)$ 時間で分かれば、1回のグラフ探索で全ての $E_1, E_2, \dots, E_{|E|}$ が求まる。以下に示すFORESTはそのようなグラフ探索アルゴリズムである。

```

Procedure FOREST; {入力: 多重グラフ $G=(V, E)$ ,
                     出力:  $E_1, E_2, \dots, E_{|E|}$ }
begin
     $E_1 := E_2 := \dots E_{|E|} := \emptyset$ ;
    各節点 $v$ のラベル $r(v)$ を0にする.
    while 未走査節点が存在すれば do
        begin
            最大ラベル $r$ の"未走査"節点 $x$ を選ぶ;
            for "未走査"辺により $x$ に隣接している各節点 $y$ に対し do
                { $E_{xy}$ 内の辺は全て未走査}
                for 各辺 $e \in E_{xy}$ に対し do
                    begin
                         $E_{r(y)+1} := E_{r(y)+1} \cup \{e\}$ ;
                         $r(y) := r(y) + 1$ ;
                        辺 $e$ を"走査済"とマークする
                    end;
                 $x$ を"走査済"とマークする
            end;
        end.

```

一般に、ある森 $T_i$ 上の2節点を結ぶ辺を新たに $T_i$ に加えられるかどうか（閉路が発生するか）という判定を $O(1)$ 時間で行うことは難しい。アルゴリズムFORESTでは、次に走査すべき節点としては、なるべく大きな添え字 $i$ を持つ $E_i$ の辺に接続するものが選ばれる。この工夫により探索された辺の両端点が同じ $T_i$ 上にあれば、常に閉路を作るという事実[12]が証明できるので、閉路判定の必要がなくなる（このために、各節点 $x$ に対し $E(x) \cup E_i \neq \emptyset$ なる最大の $i$ をラベル $r(x)$ として保持させている）。

FORESTにおいて、同一ラベル $r$ の節点を格納するためのバケットを $|E|$ 個用意し、非空バケットをリストでつないだデータ構造を用いれば、最大ラベルの未走査節点 $x$ は $O(1)$ 時間で見つかり、1回のラベルの更新も $O(1)$ 時間でできる（ラベルの変化量が常に定数1であることに注意）。従って、全体の計算時間は $O(|V| + |E|)$ で済む。算法の正当性については比較的容易に証明でき、以下の結果を得る。

【補題3.1】 [12] FORESTに $k$  ( $\geq 1$ )-辺連結グラフ $G=(V, E)$ を入力して得られる $(V, E_1)$ は $G$ の全域木であり、各 $(V, E_i)$ ,  $2 \leq i \leq |E|$ は $G - E_1 \cup E_2 \cup \dots \cup E_{i-1}$ の極大全域森である。□

【定理3.2】 [12] 連結グラフ $G=(V, E)$ に対し定理3.1を満たす辺集合 $E_i$ ,  $i=1, 2, \dots, |E|$ を $O(|E|)$ 時間で見つけることができる。ただし、 $G$ が単純グラフのとき $|E_i| \leq |V| - i$ 、 $G$ が多重グラフのとき $|E_i| \leq |V| - 1$ を満足する。□

単純グラフ  $G$  が完全グラフ  $K_{k+1}$  である場合には,  $|E_1| + |E_2| + \dots + |E_k| = k(k+1)/2$ , 多重グラフ  $G$  が  $k$  重の木であるとき,  $|E_1| + |E_2| + \dots + |E_k| = k(|V|-1)$  が満たされる.

図1の多重グラフにFORESTを適用した例を図2に示す. ただし, 図中の節点  $x_i, i=1, 2, \dots$  および辺  $e_k, k=1, 2, \dots$  はそれぞれFORESTにより  $i$  番目,  $k$  番目に走査されたことを意味する.

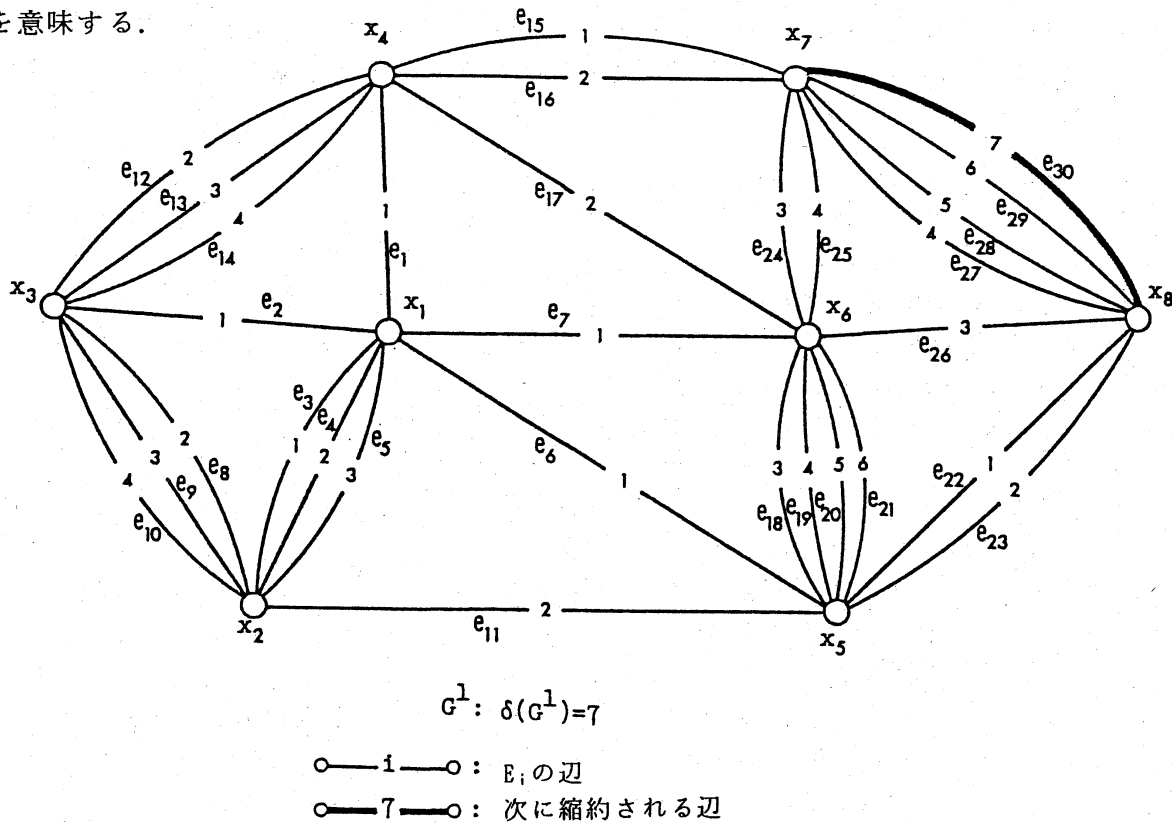


図2. 図1のグラフにFORESTを適用した例.

#### 4. $k$ -点連結全域部分グラフを求めるアルゴリズム

驚くことに, FORESTにそのまま  $k$  ( $\geq 1$ )-点連結単純グラフ  $G=(V, E)$  を入力すると, 得られる全域部分グラフ  $G'=(V, E'=E_1 \cup E_2 \cup \dots \cup E_k)$  は  $k$ -点連結になる. 正確に言えば定理3.1, 3.2に代わる次の命題が成り立つ.

【定理4.1】 [12] 単純連結グラフ  $G=(V, E)$  に対し, 各  $G_i=(V, E_1 \cup E_2 \cup \dots \cup E_i), i=1, 2, \dots, |E|$  が, 任意の  $x, y \in V$  に対し,  $\kappa(x, y; G_i) \geq \min\{i, \kappa(x, y; G)\}$ ,  $|E_i| \leq |V| - i$  を満たすような  $E$  の分割  $E_1, E_2, \dots, E_{|E|}$  を  $O(|E|)$  時間で見つけることができる.  $\square$

(実は, 文献[12]における証明を拡張することにより,  $\kappa(x, y; G)$  を  $x, y$  間の内素なパスの最大本数としたとき  $\kappa(x, y; G_i) \geq \min\{i, \kappa(x, y; G)\}$  の関係も示せる.)

FORESTにより得られた  $E_1, E_2, \dots, E_{|E|}$  に対し定理4.1が成立することの証明は, 辺連結性の場合に比べればやや複雑である. 結果を導くために用いられた次の補題を挙げておく.  $G=(V, E)$  の任意の  $E'' (\subseteq E)$  および  $G-E''$  における任意の分離節点集合  $W \subset V$  を考える.

$G-E''-W$ の1つの連結成分を $G_X=(X, E_X)$ , 残りの $G-E''-W-X$ を $G_Y=(Y, E_Y)$ で表わし,  $W$ の節点をFORESTにより走査された順に,  $s_1, s_2, \dots, s_{|W|}$ と記すと次が示せる.

【補題4.1】[12] FOREST実行中 $t$ 番目の $s_t \in W$ が走査された直後,  $\{e \in E'' \mid e \text{は走査済辺}\} \subseteq E_d \cup E_{d+1} \cup \dots \cup E_{|E|}$ なる $d \geq t+1$ が存在すれば, 次の(i), (ii)が常に成り立つ.  
 (i)  $E_i (t+1 \leq i)$ の辺で連結している $x \in X, y \in Y$ はまだ存在していない.  
 (ii)  $E_t$ の辺で連結している $x \in X, y \in Y$ は存在し得るが, その任意の $x-y$ パスは $s_t \in W$ を必ず通る.  $\square$

この補題により定理4.1は容易に導ける. 定理3.1の証明と同様に背理法を用いて $\kappa(x, y; G) \geq i+1, \kappa(x, y; G_i) = \kappa(x, y; G_{i+1}) = i$ なる $x, y, i$ を仮定する.  $E'' = E_{i+2} \cup E_{i+3} \cup \dots \cup E_{|E|}$ とすると,  $G-E''-W = G_{i+1}-W$ において $|W|=i$ なるある点カット $W$ により,  $x$ はある連結成分 $X$ に,  $y$ は残りの $Y = G-E''-W-X$ に含まれる.  $\kappa(x, y; G) \geq i+1$ より $X$ と $Y$ の間には $E''$ の辺 $e$ が存在する. これは,  $G_i$ において $e$ の両端点 $x', y'$ は連結していることを意味し, 補題3.1より $x'$ と $y'$ は $(V, E_{i+1})$ において連結している. ところが, 補題4.1より $s_{|W|}$ が走査された直後(その後永久に),  $X, Y$ は $E_h (h \geq i+1 = |W|+1)$ では連結されない. これは矛盾である.

## 5. 前処理アルゴリズムとしてのFOREST

FORESTは, グラフの連結度を保持したまま辺数を縮小でき, 必要な計算時間も線形である. 従って, グラフの連結度等を計算する際に, FORESTを前処理として実行しておけば必要な計算の手間を改良することができる.

例えば, 与えられた正整数 $k$ に対し多重グラフ $G$ が $k$ -辺連結度かどうかは $O(\min\{k, |E|^{1/2}\} |V| |E|)$ 時間で判定できることが知られているが[4], FORESTにより $G_k = (V, E_1 \cup E_2 \cup \dots \cup E_k)$ を求めた後,  $\lambda(G_k) \geq k$ の判定を行えばよいので, 計算時間を $O(|E| + \min\{k, (k|V|)^{1/2}\} k |V|^2)$ に短縮できる. また, この判定法に基づいて,  $\lambda(G)$ を効率良く決定するために,

$$\lambda(G) \geq k \text{ の判定を } k=2, 2^2, 2^3, \dots \text{ の順に } \lambda(G) < k=2^i \text{ となるまで行う. } \quad (5.1)$$

この場合, 結局 $O(|E| + \min\{\lambda(G), (\lambda(G)|V|)^{1/2}\} \lambda(G) |V|^2)$ 時間で済む[12] (これは[4]の算法では $\lambda(G) < k$ と判定されたときには既に $\lambda(G)$ が見出されていることを利用している). この計算時間は, これまでに知られている多重グラフ $G$ の $\lambda(G)$ を求める最良の $O(|E|^{3/2} |V|)$ 時間[4]に比べ短縮されている.

グラフ $G$ の点連結度 $\kappa(G)$ については,  $\kappa(G) \geq k$ を判定する $O(\max\{k, |V|^{1/2}\} k |V|^{1/2} |E|)$ 時間の算法が知られているが[6], FORESTにより上記と同様に $|E|$ を $k|V|$ に置き換えることができ,  $O(|E| + \max\{k, |V|^{1/2}\} k^2 |V|^{3/2})$ に減らせる[12].  $\kappa(G)$ の決定に必要な手間 $O(\max\{\kappa(G), |V|^{1/2}\} \kappa(G) |V|^{1/2} |E|)$ [6]についても $O(|E| + \max\{\kappa(G), |V|^{1/2}\} \kappa(G)^2 |V|^{3/2})$ 時間に減らせる[12].

グラフ $G$ 上の2点 $s, t$ 間の辺素なパスの最大本数 $\lambda$ を求める手間[4] $O(|V|^{2/3} |E|)$  ( $G$ が単純のとき) および $O(\min\{|V|, |E|^{1/2}\} |E|)$  ( $G$ が多重のとき), あるいは内素なパスの最大本数 $\kappa$ を求める手間[4] $O(|V|^{1/2} |E|)$ についても同様にそれぞれFORESTにより,  $O(|E| + \min\{\lambda, |V|^{2/3}\} |E|)$  ( $G$ が単純のとき),  $O(|E| + \min\{\lambda, |V|, \lambda^{1/2} |V|^{1/2}\} \lambda |V|)$

( $G$ が多重のとき), および  $O(|E| + \min\{\kappa, |V|^{1/2}\} \kappa |V|)$  時間に短縮できる[12]. また, これら2点間の辺素および内素なパスを見出す問題においてはFORESTにより平均的にどの程度効率が改善されるかという点についての計算機実験も行われている[10].

2部グラフ  $H=(V_1, V_2, E)$  ( $|V_1|=|V_2|$ ) 上で大きさ  $k (< |E|/|V_1|)$  のマッチングの実現するために, 定理4.1を使って次のように予めグラフの辺数を  $O(k|V_1|)$  に縮小することができる ( $H$ の最大マッチングの大きさ  $\alpha$  が  $\alpha \geq |E|/|V_1|$  を満たすことは大きさ  $\alpha$  の最小節点被覆が存在することから導けるので, 大きさ  $k < |E|/|V_1|$  のマッチングの存在は自明である).  $H$ に新たに2点  $s, t$  および  $s$  と  $V_1$  間,  $t$  と  $V_2$  間を結ぶ辺を加えたグラフ  $\bar{H}=(V_1 \cup V_2 \cup \{s, t\}, \bar{E})$  にFORESTを適用し  $\bar{H}_k=(V_1 \cup V_2 \cup \{s, t\}, \bar{E}_1 \cup \bar{E}_2 \cup \dots \cup \bar{E}_k)$  を見出す. 定理4.1より2部グラフ  $\bar{H}_k - \{s, t\}$  には大きさ  $k$  のマッチングが存在しており, 辺数は  $2k|V_1|$  以下である (しかし, 残念ながら最大マッチングなど, 大きさ  $k \geq |E|/|V_1|$  のマッチング実現のためにはこのやり方では辺数のオーダーを減らせない).

この他に, FORESTにより計算時間の短縮されるグラフアルゴリズムとして6-点連結グラフ  $G=(V, E)$  上で2組の2点対間の点素なパス (常に存在する) を見出す  $O(|V||E|)$  時間の算法[18]が挙げられる. これは, 明らかに定理4.1により  $O(|E| + |V|^2)$  に短縮される.

## 6. 多重グラフの辺連結度を求めるアルゴリズム

現在, 単純グラフ  $G=(V, E)$  の辺連結度  $\lambda(G)$  は  $O(\lambda(G)|V|^2)$  時間で求まることが知られている[11]. グラフが多重の場合の  $\lambda(G)$  の計算時間は, 前節で議論したようにFORESTを前処理として用いることにより  $O(|E| + \min\{\lambda(G), (\lambda(G)|V|)^{1/2}\} \lambda(G)|V|^2)$  で済むが, 本節ではFORESTの持つ有用な性質を利用して, より高速な  $O(|E| + \lambda(G)|V|^2)$  時間アルゴリズムを開発する (この計算の手間は次節でさらに短縮される).

定理3.1の条件を満たす  $E$  の分割  $E_1, E_2, \dots, E_{|E|}$  において, ある辺  $e$  が  $E_h$  に含まれるとき, 各  $i \leq h$  に対し  $e$  の両端  $u, v$  を結ぶ  $E_i$  の辺から成るパスが全域森  $(V, E_i)$  上に存在することは容易に分かる (例えば図2の適用例を見よ). これより次を得る.

【補題6.1】 [12] 定理3.1の条件を満たす  $E$  の分割  $E_1, E_2, \dots, E_{|E|}$  が見出されているとき, 任意の辺  $e=(x, y) \in E_i$  に対し,  $\lambda(x, y; G) \geq i$  であり,  $x, y$  間の  $\lambda(x, y; G)$  本の辺素なパスは  $O(|V| + |E|)$  時間で見つかる.  $\square$

定理3.1の条件を満たす  $E$  の分割  $E_1, E_2, \dots, E_{|E|}$  のうちをFORESTにより見出されたものは, 特に以下の有用な性質を持つ. FORESTにより最後に走査された節点  $t$  については,  $t$  に接続する辺  $E(t)$  は全て  $t$  に隣接する節点の走査中に走査され, その度に  $t$  のラベル  $r(t)$  が1ずつ増加していく. 従って,  $t$  に接続する各辺は常に全て異なる分割  $E_i$  に属する, すなわち,  $|E(t) \cap E_i| = 1, i=1, 2, \dots, |E(t)|$  が成り立つ (例えば, 図2の  $x_8$  を見よ).  $E(t) \cap E_{|E(t)|}$  の辺を  $e=(s, t)$  とすると補題3.1より  $\lambda(s, t; G) \geq |E(t)|$ . しかし,  $E(t)$  は  $s, t$  を分離するカットであるので  $\lambda(s, t; G) = |E(t)|$  を得る. これは次を意味する.

【補題6.2】 任意の無向多重グラフ  $G$  において  $\lambda(x, y; G) = \min\{|E(x)|, |E(y)|\}$  ( $\geq \delta(G)$ ) なる辺  $e=(x, y)$  が必ず存在する. また, そのような辺はFORESTにより線形時間で見出せる.  $\square$

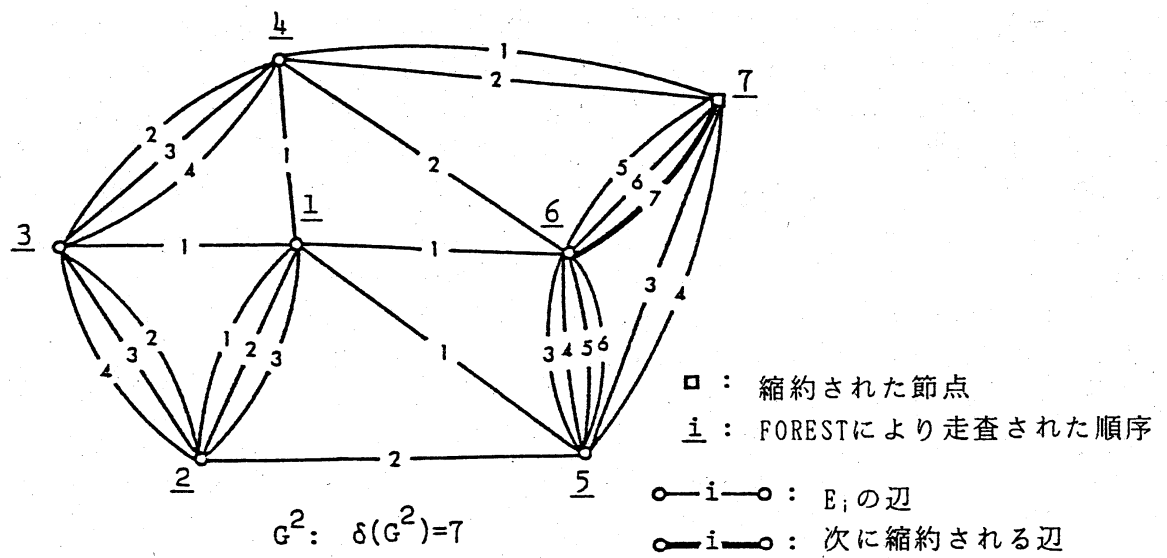
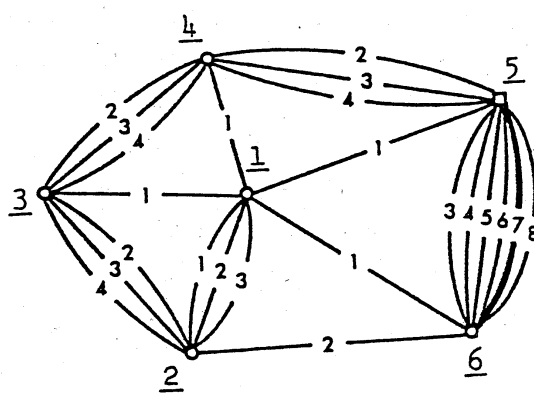
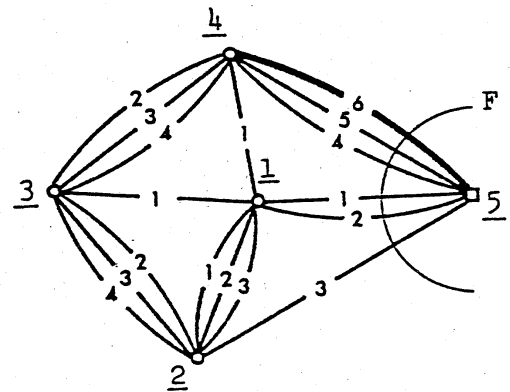
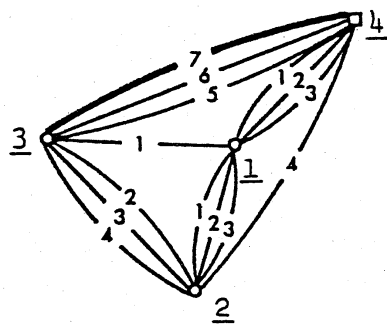
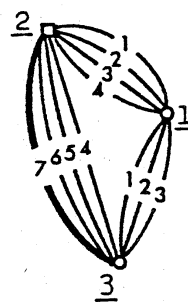
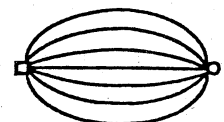


図 3. TESTECの適用例.

図 4.  $G^3: \delta(G^3)=7$ 図 5.  $G^4: \delta(G^4)=6$ 図 6.  $G^5: \delta(G^5)=7$ 図 7.  $G^6: \delta(G^6)=7$ 図 8.  $G^7: \delta(G^7)=7$



ところで、一般に多重グラフ  $G=(V, E)$  ( $|V| \geq 2$ ) の任意の 2 点  $u, v$  を縮約しても辺連結度は減少しない。すなわち、 $\lambda(G/\{u, v\}) \geq \lambda(G)$ 。逆に、 $\lambda(u, v; G) \geq k$  なる  $k$  に対しては、 $\lambda(G/\{u, v\}) \geq k$  ならば  $\lambda(G) \geq k$  が成り立つ。次の補題はこれらの性質から容易に導くことができる。

【補題 6.2】 [13]  $|V| \geq 3$  のとき、 $\lambda(u, v; G) \geq \delta(G)$  なる  $u, v$  に対し、  
 $\lambda(G) = \min \{ \delta(G), \lambda(G/\{u, v\}) \}$ 。□

補題 6.2, 6.3 より  $G=(V, E)$  の  $\lambda(G)$  を求めるには以下のようにすればよい： $G^1 := G$  とし、FOREST により  $\lambda(u, v; G) \geq \delta(G)$  なる  $u, v$  を見出し、 $G^2 := G^1/\{u, v\}$  とする。以下、節点数が 2 になるまで、 $G^3, G^4, \dots, G^{|V|-1}$  を同様に構成した後、 $\lambda(G)$  は  $\min\{\delta(G^1), \delta(G^2), \dots, \delta(G^{|V|-1})\}$  で与えられる。

図 1 のグラフにこのアルゴリズム（以下、TESTEC と呼ぼう）を適用した例を図 2 ～ 8 に示す（この結果、 $\lambda(G) = 6$  であることが判明する。また、図 5 の  $F \subseteq E$  は  $\lambda(G)$  を与える  $G$  の 1 つの最小カットである）。

アルゴリズム TESTEC の計算時間は  $O(|V||E|)$  で済むことは FOREST の計算時間から容易に分かるが、(5.1) と同様の工夫を行うことにより、これを  $O(|E| + \lambda(G)|V|^2)$  時間に減らすことができる [13]（従って、 $G$  が単純の場合にはこの手間は [11] の計算時間に一致する）。

## 7. 実数容量付き無向ネットワークの最小容量カットを求めるアルゴリズム

本節では、無向グラフ  $G$  の各辺  $e$  に正值の容量  $c(e)$  を持つネットワーク  $N=(G, c)$  を考え、最小容量カット値  $c(N) \equiv \min \{ \sum_{e \in F} c(e) \mid F \text{ は } G-F \text{ が非連結となる } E \text{ の部分集合} \}$  を見出す効率の良いアルゴリズムを与える。

容易に分かることだが、 $c(N)$  の値は、 $N$  の  $|V|(|V|-1)/2$  組の各 2 点対間の最大流の流量値のうちの最も小さい値と一致する。Gomory と Hu [9] の結果によれば  $N$  の全ての 2 点対間の最大流の値は  $|V|-1$  回最大流問題を解けば得られるので、 $c(N)$  は少なくとも  $O(|V|T(|V|, |E|))$  時間で求まる。ただし、 $T(|V|, |E|)$  は最大流を計算するための手間で、現在、強多項式時間では  $O(|V||E|\log(|V|^2/|E|))$  が最良である [8]。以下では、 $c(N)$  を計算する  $O(|V||E| + |V|^2 \log |V|)$  時間のアルゴリズムを与える。

まず、ネットワーク  $N=(G, c)$  の容量が全て整数値である場合を考えよう。 $N$  の各辺  $e$  を  $c(e)$  本の多重辺に置き換えて得られるグラフ  $\bar{G}=(V, \bar{E})$  に対し、 $\lambda(\bar{G}) = c(N)$  であるのは明らかである。従って、前節のアルゴリズム TESTEC を用いれば、 $c(N)$  は  $O(|\bar{E}| + c(N)|V|^2) = O(\sum c(e) + c(N)|V|^2)$  時間で求まる。しかし、この計算の手間は  $\log c$  の多項式ではない。この原因は FOREST の実行に  $O(|\bar{E}|)$  時間要するためである。

この点を改善するために、FOREST を直接ネットワーク  $N=(G=(V, E), c)$  に適用しても正しく作動するよう修正する。 $\bar{G}=(V, \bar{E})$  を仮想しながら  $N$  に FOREST を適用してみよう。 $N$  のある未走査辺  $e=(x, y)$  ( $\bar{G}$  では  $|\bar{E}_{xy}| = c(e)$  なる未走査辺集合) を走査する際、 $\bar{G}$  では  $\bar{E}_{xy}$  内の全ての辺は連続して走査されるが、 $N$  に対するこの処理を単位時間で行うために、 $\bar{E}_{xy} \cap E_i \neq \emptyset$  なる最大の  $i$  だけを  $q(e)$  として保持しておく ( $c(e) = |\bar{E}_{xy}|$  を記憶しておけば、得られた  $q(e), e \in E$  から  $\bar{E}$  の分割  $\bar{E}_1, \bar{E}_2, \dots$  を求めることもできる)。このように FOREST

を容量付きネットワークNが入力できるように修正したアルゴリズムCAPFORESTを以下に記述しておく.

```

Procedure CAPFOREST; {入力:  $N=(G=(V,E),c)$ , 出力:  $q(e), e \in E$ }
begin
  全ての節点  $v \in V$  に対して  $r(v) := 0$ ;
  全ての辺  $e \in E$  に対して  $q(e) := 0$ ;
  while 未走査節点が存在すれば do
    begin
      最大ラベル  $r$  の "未走査" 節点  $x$  を選ぶ;
      for "未走査" 辺により  $x$  に隣接している各節点  $y$  に対し do
         $\{E_{xy}$  内の辺は全て未走査 $\}$ 

        for 各辺  $e \in E_{xy}$  に対し do
          begin
             $q(e) := r(y) + c(e)$ ;
             $r(y) := r(y) + c(e)$ ;
            辺  $e$  を "走査済" とマークする
          end;
         $x$  を "走査済" とマークする
      end;
    end.

```

このCAPFORESTの計算時間は未走査節点の集合をフィボナッチヒープ[5]で管理しておくことにより  $O(|E| + |V| \log |V|)$  に抑えることができる[13].

辺連結度を求めるアルゴリズムTESTECに対してもNに直接適用できるよう同様の修正を施してやれば, CAPFORESTを  $|V|-1$  回用いて整数値ネットワークNの辺連結度に相当する  $c(N)$  を計算するアルゴリズムMINCUTが次のように得られる[13]. ただし,  $\delta(N) \equiv \min \{ \sum_{e \in E(v)} c(e) \mid v \in V \}$  とする.

```

Procedure MINCUT; {入力:  $N=(G=(V,E),c)$ , 出力:  $c(N)$  および最小カット  $F$ }
begin
   $N' := N$ ;  $k := +\infty$ ;
  while  $N' = (G' = (V', E'), c')$  において  $|V'| \geq 3$  なら do
    begin
       $N'$  にCAPFORESTを適用し  $q(e), e \in E'$  を見出す;
       $\sum_{e \in E'(w)} c(e) = \delta(N')$  なる節点  $w \in V'$  を選ぶ;
      if  $\delta(N') < k$  then  $F := E'(w)$ ;
       $k := \min\{k, \delta(N')\}$ ;
       $q(e) \geq \delta(N')$  なる辺  $e = (u, v) \in E'$  を選び  $G' := G' / \{u, v\}$ ;
       $N' := (G', c')$ 
    end

```

```

end;
{|V'|=2より  $c(N') = \delta(N')$ }
if  $\delta(N') < k$  then  $F := E'$ ;
 $k := \min\{k, \delta(N')\}$ ;
 $c(N) = k$  および  $F$  は  $N$  の最小カットと結論する
end.

```

明らかに、アルゴリズム MINCUT の計算時間は  $O(|V||E| + |V|^2 \log |V|)$  で済む。MINCUT が有理数値容量を持つネットワークに対しても正常に作動することは容易に分かるであろう。さらに、容量が一般の実数であっても正しく答えを出力することも有理数近似の考え方を用いて証明されている [13]。

MINCUT の結果を用いると前節の  $\lambda(G)$  を求める手間を若干改善することができる。与えられたグラフ  $G=(V, E)$  から、各 2 点間の多重辺  $E_{uv}$  を  $c(e) = |E_{uv}|$  なる 1 本の辺  $e=(u, v)$  に置き換えて得られる容量付きネットワーク  $N_G$  のに対し MINCUT を適用してやれば、 $O(|E| + p|V| + |V|^2 \log |V|)$  時間で  $\lambda(G) = c(N_G)$  が得られる（ただし、 $p$  は間に辺の存在する点対の総数である）。TESTEC と MINCUT をうまく切り替えて  $\lambda(G)$  を見いだす  $O(|E| + \min\{\lambda(G)|V|^2, p|V| + |V|^2 \log |V|\})$  時間のアルゴリズムを構成することができる [13]。

## 8. グラフの最小カットの総数を数えるアルゴリズム

グラフ  $G$  の辺連結度  $\lambda(G)$  に対して明らかに、 $C^k(G) = \emptyset$  ( $k < \lambda(G)$  のとき)、 $C^k(G) \neq \emptyset$  ( $\lambda(G) \leq k \leq |E|$  のとき) が成立するが、グラフの持つ最小カットの総数  $|C^{\lambda(G)}|$  を計算することは、辺の故障率を導入した確率グラフの連結確率の近似計算 [16] や super- $\lambda$  と呼ばれるグラフの信頼性の検証 [19, 20] を行う際に必要となる。これまでに 2 点  $s, t$  を分離する  $s$ - $t$  最小カットの総数  $\mu = |C^k(s, t; G)|$  ( $k = \lambda(s, t; G)$ ) を計算する  $O(|E||V| + \mu|E|)$  時間アルゴリズムやグラフの最小カットの総数  $|C^k(G)|$  ( $k = \lambda(G)$ ) を求める  $O(|V|^2|E|)$  時間のアルゴリズムが知られている [1]。

本節では、FOREST の性質に基づいてこれら 2 つのアルゴリズムの計算時間が短縮できることを示す。定理 3.1 の条件を満たすグラフ  $G$  の辺集合  $E$  の分割  $E_1, E_2, \dots, E_{|E|}$  に対して次が成り立つことは定理 3.1 と類似の議論により証明できる。

【補題 8.1】 [14]  $G_i = (V, E_1 \cup E_2 \cup \dots \cup E_i)$  としたとき、任意の 2 節点  $x, y$  および  $\lambda(x, y; G) < i \leq |E|$  なる  $i$  に対し、 $C^k(x, y; G_i) = C^k(x, y; G)$  ( $k = \lambda(x, y; G)$ ) が成り立つ。□

Ball と Provan による  $s$ - $t$  最小カット数  $\mu$  を数えるアルゴリズムは、 $s, t$  間の辺素なパスを最大本数求めた後、このパス集合に基づいて  $s$ - $t$  最小カットを 1 つずつ線形時間で枚挙していくのもので、その手間  $O(|E||V| + \mu|E|)$  は、 $s, t$  の最大本数の辺素なパスを見つける手間を  $T$  とすると正確には  $O(T + \mu|E|)$  と表わせる [1]。従って、 $T$  には 5 節の結果を用い、グラフを補題 8.1 より  $G_{k+1}$  ( $k = \lambda(s, t; G)$ ) に縮小しておけば、彼らのアルゴリズムの手間は、 $O(|E| + \min\{\lambda(s, t; G), |V|^{2/3}\}m')$  (グラフが単純のとき) および  $O(|E| + \min\{|V|, \lambda(s, t; G), \sqrt{m'}\}m')$  (グラフが多重のとき) に改善できる。ただし、 $m' = \min\{|E|, k|V|\}$ 。グラフの最小カットの集合の総数  $|C^k(G)|$  ( $k = \lambda(G)$ ) を計算する

BallとProvanのアルゴリズムは次の容易に示せる事実に基づいている。すなわち、 $|C^k(G)|$  ( $k=\lambda(G)$ ) は、 $|C^k(G/\{u,v\})|$  ( $\lambda(u,v;G)>k$  のとき)、あるいは  $|C^k(u,v;G)| + |C^k(G/\{u,v\})|$  ( $\lambda(u,v;G)=k$  のとき) に等しい。従って、 $|C^k(G)|$  を求めるには、まず  $\lambda(G)$  を決定した後 (これは  $O(|V|T)$  時間で済む [4, 17])、任意の 2 点  $u, v$  間の局所辺連結度  $\lambda(u,v;G)$  を調べ、 $\lambda(u,v;G) > \lambda(G)$  なら  $u, v$  を縮約し、そうでなければ  $|C^k(u,v;G)|$  を計算した後  $u, v$  を縮約するという操作を繰り返せば良い。このとき要する手間は、 $u-v$  最小カットが 1 つ当たり線形時間で見出せることから  $O(|V|T + |C^k(G)||E|)$  で済む。ここで、 $|C^k(G)|$  が  $|V|, |E|$  の指数関数になる心配があるが、Bixby[2]により常に  $O(|V|^2)$  で抑えられることが知られている。この事実と文献[4]の  $T \leq O(|V||E|)$  の結果から Ball と Provan は当時  $O(|V|^2|E|)$  時間のアルゴリズムを与えたことになる。

この計算時間を FOREST により改良するポイントは以下の 2 つである。

- (1) 補題 8.1 によりグラフを縮小しておく。
- (2) 補題 6.2 を用いる (すなわち、指定した 2 点  $u, v$  間の辺素なパスを求めるのではなく、FOREST により線形時間で補題 6.2 を満たす 2 点見出す)。

(1), (2) の工夫により  $|C^k(G)|$  ( $k=\lambda(G)$ ) の計算は  $O(|E| + \lambda(G)|V|^2 + \lambda(G)|C^k(G)||V|)$  時間に短縮できる [14]。

## 9. おわりに

グラフ探索アルゴリズムには、深さ優先探索、幅優先探索がよく知られており、これらは 2 点連結性や距離レベルなどのグラフの有用かつ基本的な性質を線形時間で見出す重要な役割を持っている。本稿で与えたアルゴリズム FOREST もグラフ探索の 1 つであり、主にグラフの連結度を調べるのに役立つ基本的な算法と考えられる。今後もこのアルゴリズムの利用可能な問題を調べていく予定である。

**謝辞** 日頃温かい御指導を賜る豊橋技術科学大学楠菊信教授に感謝致します。本研究は一部文部省科学研究費および中部電力助成金の援助のよる。

## 文 献

- [1] M.O. Ball and J.S. Provan: "Calculating bounds on reachability and connectedness in stochastic networks", Networks, Vol.13 (1983) 253-278.
- [2] R.E. Bixby: "The minimum number of edges and vertices in a graph with edge connectivity  $n$  and  $m$   $n$ -bond", Networks, Vol.5 (1975) 253-298.
- [3] F.T. Boesch: "Synthesis of reliable networks - A survey", IEEE Trans. Reliability, Vol. R-35 (1986) 240-246.
- [4] S. Even and R.E. Tarjan: "Network flow and testing graph connectivity", SIAM J. Computing 4 (1975) 507-518.
- [5] M.L. Fredman and R.E. Tarjan: "Fibonacci heaps and their uses in improved optimization algorithms", 25th Symp. Found. Compt. Sci. (1984) 338-346.
- [6] Z. Galil: "Finding the vertex connectivity of graphs", SIAM J. Computing 9

(1980) 197-199.

- [7] M.R.Garey and D.S.Jhonson: Computers and Intractability, A Guide to the Theory of NP-completeness, Freeman, W.H. and Company, 1979.
- [8] A.V.Goldberg and R.E.Tarjan: "A new approach to the maximum flow problem", Proc. 18th ACM Symp. on the Theory of Comput. (1986) 136-146.
- [9] R.E.Gomory and T.C.Hu: "Multi-terminal network flows", J.SIAM Vol.9 (1961) 551-570.
- [10] 近藤 健次: "k-連結全域部分グラフを求めるアルゴリズム (FOREST) の性能の検討", 豊橋技術科学大学情報工学科, 卒業論文 (1990)
- [11] D.W.Matula: "Determining edge connectivity in  $O(nm)$ ", Annu. Symp. Found. Compt. Sci. 28th (1987) 249-251.
- [12] H.Nagamochi and T.Ibaraki: "Linear time algorithms for finding k-edge-connected and k-node-connected spanning subgraphs", Technical Report #89006, Dept. of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University (1989).
- [13] H.Nagamochi and T.Ibaraki: "Computing edge-connectivity in multiple and capacitated graphs", Technical Report #89009, Dept. of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University (1989).
- [14] H.Nagamochi, Z.Sun and T.Ibaraki: "Counting the number of minimum cuts in multiple undirected graphs", Technical Report #89010, Dept. of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University (1989).
- [15] T.Nishizeki and S.Poljak: "Highly connected factors with a small number of edges", Working paper (1989).
- [16] J.S.Provan: "Bounds on the reliability of networks", IEEE Trans. Reliability, Vol.R-35 (1986) 260-268.
- [17] C.P.Schnorr: "Bottlenecks and edge connectivity in unsymmetrical networks", SIAM J. Computing Vol.8 (1979) 265-274.
- [18] Y.Shiloach: "A polynomial solution to the undirected two paths problem", J. of the Association for Computing Machinery, Vol.27 (1980) 445-456.
- [19] 孫, 永持, 楠: "与えられた基数のカットセット数を最小にするグラフ", 信学論, Vol.J72-A (1989) 1601-1610.
- [20] Z.Sun, H.Nagamochi and K.Kusunoki: "Multiple graphs minimizing the number of minimum cut-sets", 信学論 (submitted).
- [21] 鈴木, 高橋, 西関: "3-連結全域部分グラフを求めるアルゴリズム", 情報処理学会研究報告 89-AL-7-2 (1989).